САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ
ЭНЕРГЕТИЧЕСКИХ СИСТЕМ

Пузач Владислав Александрович

Выпускная квалификационная работа бакалавра

Оптимизация режима работы светофоров

Направление 01.03.02
«Прикладная математика и информатика»

Научный руководитель,
доктор физ.-мат. наук,
профессор
Крылатов А.Ю.

Санкт-Петербург
2020

Содержание

Введение

Развитие транспортных средств всегда преследовало следующие цели: повышение скорости передвижения, грузоподъёмности, надёжности и уровня комфорта. Появление в начале 20 века автомобилей, последовательное улучшение их характеристик и массовая доступность привели к тому, что теперь крупная часть населения имеет возможность обладать транспортным средством, удовлетворяющим самым высоким стандартам.

Значительное повышение спроса на автомобили привело к ряду проблем. Высокое количество выбросов вредных веществ, активное использование невозобновляемых ресурсов и, конечно, коллапсы транспортной системы существенно влияют на жизнь современного человека в негативном ключе.

Дорожные пробки ведут к экономическим потерям, ухудшению экологической ситуации, оказывают негативное воздействие на психическое и физическое здоровье человека. Крупные временные затраты и дискомфорт, вызванные дорожными заторами, противоречат начальным целям использования автомобиля. Главными причинами возникновения коллапсов являются: высокий спрос, низкие пропускные способности дорог, отсутствие достаточного количества различных маршрутов, неграмотная градостроительная политика, ремонтные работы в неподходящее время, долгий процесс оформления ДТП и ликвидации последствий.

Для решения данной проблемы было предложено много способов. Дорожно-строительные: увеличение габаритов проезжей части, наём больших бригад, строительство дополнительных магистралей. Требуют высоких затрат, при неграмотном планировании могут привести даже к ухудшению ситуации. Административные: изменение разметки, установка знаков, настройка светофоров. Самые дешёвые, однако при некоторых условиях их значение может быть несущественным. Социальные: пропаганда ЗОЖ,

призывы к более уважительному стилю вождения, просьбы не ездить в одиночку и пользоваться общественным транспортом. Существуют и противодействующие силы, такие как личный выигрыш во времени, уровне комфорта, при этом поведение водителей является значимым фактором только при высоком уровне развития транспортной системы.

Одним из методов, не требующих большого количества материальных затрат, однако обещающего достаточно высокую эффективность, является управление сигналами светофора.

Используемые определения

Фаза светофора – промежуток времени, в течение которого движение на перекрёстке разрешено только одному из направлений.

Цикл светофора – совокупность всех фаз светофора.

Разбиением цикла светофора называется набор длительностей его фаз. Если на перекрёстке имеются всего два направления, то разбиением называется отношение длительности одной из фаз к длине всего цикла.

Смещение фазы – число, которое показывает, насколько должен быть смещён момент начала зелёной фазы относительно некоторого главного светофора на этой улице.

Матрица корреспонденций – матрица, элементами которой являются объёмы перемещений между районами отправления и прибытия за некоторый промежуток времени.

Постановка задачи

Управление может принадлежать к одному из трёх типов: фиксированное,реагирующее, адаптивное. Фиксированное управление не зависит от текущей ситуации на дороге, является легко реализуемым и самым широко распространенным. Второй тип даёт различные настройки светофора для различных дорожных обстановок, однако некоторые параметры, такие как длина цикла, смещение фазы остаются постоянными. Адаптивное управление позволяет варьировать все параметры настройки сигналов.

Объектом управления можно выбрать изолированное пересечение или их систему. При рассмотрении системного уровня возможно учесть большее количество факторов, влияющих на движение, следовательно такой подход может быть более продуктивным и более сложным.

Целью данной работы является построение реагирующего управления сигналами светофоров для системы пересечений.

Временной период p управления в течение каждого дня состоит из M циклов. Пусть цикла под номером t: $D_{p,t}$ – матрица корреспонденций для транспортной сети , $L_{p,t}$ – матрица настроек сигналов светофоров, $x_{p,t+1} = \phi_t(D_{p,t}, L_{p,t})$ – вектор длин очередей на перекрёстках. Обозначим за i некоторый пункт отправления, а за j пункт, на котором расположен светофор. Тогда $L^{ij}_{p,t}$ = разбиение цикла светофора по направлению из i в j. Таким образом, формально задачу можно поставить следующим образом: необходимо в каждый момент (p, t), зная $D_{p,t}$, $L_{p,t-1}$, ..., $L_{p,t-s}$, $x_{p,t-1}$, ..., $x_{p,t-m}$, получить такое $\bar{L}_{p,t}$, что:

$$\bar{L} = \underset{L_0,\ldots,L_{M-1}}{\operatorname{argmin}} \; E[\sum_{t=0}^{M-1} (x^T_{p,t+1}\, x_{p,t+1})]$$

где математическое ожидание берётся по вектору $(D_{p,0}, ..., D_{p,M-1})$, обла-

дающим неизвестным распределением.

Оптимальное решение для представленного критерия минимизирует длины очередей в течение одного дня(одного периода управления). Квадратичная функция используется для акцентирования в первую очередь на сокращение особенно длинных очередей, жертвуя некоторым увеличением коротких. Все векторы очередей для различных номеров циклов входят в сумму равноправно, указывая на отсутствие исключительной важности отдельных циклов внутри периода.

Решение должно:

1. показать повышенную эффективность в сравнении с фиксированной стратегией
2. не требовать много времени для обучения
3. не использовать затратных в вычислительном плане процедур
4. опираться на современные концепции и методы

Обзор литературы

Фиксированное управление, основанное на заранее определённых длине и разбиении цикла, может использоваться при относительно стабильных и регулярных потоках. Работа [6] описывает широко распространённый метод поиска фиксированного управления TRANSYT, который может использоваться для составления стратегий для нескольких часов в течение дня. Так как транспортная система представляет собой динамическую систему, ни одно постоянное разбиение цикла не может отвечать наилучшим образом всем реальным условиям движения. После возникновения различных технических средств, позволяющих измерять объёмы транспортных потоков, появляются новые подходы, целью которых является построение управления, работающего в реальном времени на основе полученных данных.

В Австралии активно используется подход SCATS, предложенный в [14]. Данный подход имеет несколько недостатков: не адаптируется к быстро меняющимся условиям движения, не оптимизирует эффективности всей сети, требует ручной настройки смещений. Статья [9] посвящена описанию метода SCOOT. В работе [8] авторы проводят анализ данного метода и приходят к выводу, что SCOOT показывает хорошие результаты для зонного контроля, но не справляется с общесистемным. Примером системного контроля является REALBAND [7], определяющий группы средств, двигающихся вместе. Однако данный подход ограничен для использования типами транспортных условий, не предполагающих дорожных заторов, для которых трудно выделить подобные группы.

На транспортные системы влияет множество факторов: инфраструктура, погода, типы транспортных средств, пешеходы и т.д. Каждый фактор имеет оригинальные особенности, которые делают транспортную систему нелинейной стохастической. Даже учитывая тот факт, что упомянутые вы-

ше подходы развивались и применяются с начала 80-х годов, проблема всё ещё не решена. Использование методов вычислительного интеллекта позволяет избежать скрупулёзного подбора точной модели, адаптировать управление под неизбежные изменения динамики системы.

Q-обучение использовалось для оптимального управления изолированным перекрёстком в [2], [20], для системы перекрёстков в [19]. Однако матрица пар состояние-действие требует больших затрат памяти, что ограничивает применение этого метода для систем с большим набором принимаемых состояний.

Спалл и Чен в [18] предложили подход S-TRAC(System-Wide Traffic-Adaptive Control). Эта работа примечательна тем, что стратегия управления не предполагает использования модели транспортной системы. Подход основан на использовании нейронной сети, обученной с помощью алгоритма SPSA, разработанного Спаллом в [17]. В [3] смоделировано движение внутри центральной части Манхэттэна, что было использовано в [18]. В [4] используется подход из [18], но для другой транспортной системы и меры эффективности. Результаты применения подхода для смоделированных данных выглядят многообещающе, однако [18] не содержит строгого обоснования сходимости процедуры для любых условий движения. В ВКР работоспособность данного подхода проверяется в главе 3 для модели из статьи [1].

SPSA относится к классу алгоритмов стохастической аппроксимации, детальное описание которых представлено в [25]. Ускоренная версия алгоритма, представляющая собой модификацию известного метода Ньютона-Рафсона, рассматривается в [15]. Продолжением работы [15] является статья [16], содержащая много ценных замечаний насчёт правильного использования ускоренной процедуры. В подходе из [18] SPSA играет решающую роль, поэтому глава 1 данной ВКР посвящена обсуждению различных алгоритмов стохастической аппроксимации.

Статья [5] описывает схему применения SPSA для прямого обучения нейронной сети в задачах управления, также в данной работе предложена сглаживаемая версия данного алгоритма. [5] послужила мотивацией для ознакомления с концепцией прямого обучения нейронной сети, краткий обзор которой содержит глава 2.

Глава 1

Стохастическая аппроксимация

1.1 Постановка задачи, метод стохастической аппроксимации

Имеется набор $\{\omega^n\} \subset \mathbb{R}^p$ из некоторого распределения $P_\omega(\cdot)$. Функцией средних потерь назовём:

$$f(l) = \int_{\mathbb{R}^p} F(\omega, z) P_\omega(d\omega) \tag{1.1}$$

где $F(\omega, z)$: $\mathbb{R}^p \times \mathbb{R}^r \longrightarrow \mathbb{R}$ – функция потерь. Согласно [25]: "***методом стохастической аппроксимации*** *называют последовательный способ улучшения оценки, минимизирующей функционал среднего риска (1.1), использующий на каждом шаге новые наблюдения и предшествующую оценку*". Предполагая дифференцируемость $F(\omega, z)$ по z, можно найти все оптимальные вектора из соотношения:

$$g(z) = \int_{\mathbb{R}^r} \bigtriangledown_z F(\omega, z) P_\omega(d\omega) = 0$$

Будем считать, что вид вероятностной меры $P_\omega(\cdot)$ не является доступным, имеется соответствующая этой мере выборка ω^1, ω^2, ... и представляется возможным наблюдать y_n, которые при задании z^n являются или $F(\omega^n, z^n)$, или $\bigtriangledown_l F(\omega^n, z^n)$, быть может, с помехами. В качестве алгоритма нахожде-

ния θ, удовлетворяющего условию (1.1), возможно применять следующую рекурсию:

$$\hat{\theta}^k = \hat{\theta}^{k-1} - \alpha_k \, \hat{g}^k(\hat{\theta}^{k-1})$$

в которой $\{\alpha_k\}$ – определяемая экспериментатором последовательность коэффициентов, причём $\alpha_k \geq 0 \, \forall k$; $\hat{g}^k(\hat{\theta}^{k-1})$ – некоторая построенная оценка градиента $f(\cdot)$.

1.2 Алгоритм Роббинса-Монро

В работе [12] был предложен первый стохастический алгоритм. Перед исследователями стояла проблема поиска действительного корня z неизвестной вещественнозначной функции $g(z)$, зашумленные значения которой являются доступными для измерения в заданных точках.

Предполагается, что $\forall z \in \mathbb{R}$ сопоставлена действительная с. в. $G(\omega, z)$ с неизвестным распределением и мат. ожиданием:

$$f(z) = E_\omega\{G(\omega, z)\} = \int_{-\infty}^{+\infty} G(\omega, z) P_\omega(d\omega)$$

Необходимо используя выборку $G(\omega^1, z^1)$, $G(\omega^2, z^2)$, ... определить точку θ, для которой: $f(\theta) = 0$. Процедура Роббинса-Монро имеет следующий вид:

$$\hat{\theta}^k = \hat{\theta}^{k-1} - \alpha_k Y_k$$

$$\alpha_k \xrightarrow{k\to\infty} 0, \sum_{k=1}^{\infty} \alpha_k = \infty, \sum_{k=1}^{\infty} \alpha_k^2 < \infty$$

$$Y_k = G(\omega^k, \hat{\theta}^{k-1}) + \upsilon_k$$

где υ_k – шум при k-ом измерении.

Теоремы, показывающие состоятельность оценок при различных условиях, приведены в [25].

Данный алгоритм применим и для случая функции нескольких переменных.

1.3 Процедура Кифера-Вольфица

Ставится задача минизации функционала среднего риска (1.1). При поиске точки минимума вид $f(\cdot)$, $F(\cdot,\cdot)$ считается неизвестным, а последовательность $\{\omega^k\}$ – неконтролируемой.

В статье [11] для решения задачи применяли следующий алгоритм:

$$\hat{\theta}^k = \hat{\theta}^{k-1} - \alpha_k \frac{Y_k^+ - Y_k^-}{2\beta_k}$$

при этом

$$Y_k^{\pm} = \begin{pmatrix} F(\omega^{2r(n-1)+\frac{3\pm1}{2}}, \hat{\theta}^{k-1} \pm \beta_k e^1 \\ F(\omega^{2r(n-1)+\frac{7\pm1}{2}}, \hat{\theta}^{k-1} \pm \beta_k e^2 \\ \cdots \\ F(\omega^{2r(n-1)+\frac{1\mp1}{2}}, \hat{\theta}^{k-1} \pm \beta_k e^r \end{pmatrix}$$

Авторы работы [11] доказали состоятельность оценок при определённых допущениях о распределении $\{\omega_k\}$, свойствах $F(\cdot,\cdot)$ и детерминированных последовательностей $\{\alpha_k\}$, $\{\beta_k\}$.

Недостатки алгоритма:

1. каждая итерация требует выполнения 2r измерений, что часто является слишком затратным условием для реальной задачи
2. предполагаются серьёзные ограничения на $\{\omega_k\}$
3. при почти произвольных помехах состоятельность оценок не получается

1.4 Рандомизированные алгоритмы

1.4.1 Алгоритмы первого порядка

В процедуре Кифера-Вольфица исследование значений функции идёт в соответствии с направлениями $e^1, ..., e^r$. Позднее в [23] вместо фиксированных направлений было предложено использовать пробное возмущение при новых измерениях. Причём возмущению подвергаются все управляемые параметры, а не только один.

Обозначим за $\{\Delta_k\}$ последовательность одинаково симметрично распределённых векторов с матрицей ковариаций

$$cov\{\Delta_k \Delta_j^T\} = \delta_{kj}\sigma_\Delta^2 I$$

и ограниченным вторым статистическим моментом. Вместо процедуры Кифера-Вольфица можно воспользоваться одним из следующих методом с двумя измерениями:

$$\begin{cases} z^{2k} = \hat{\theta}^{k-1} + \beta_k * \Delta_k \\ z^{2k-1} = \hat{\theta}^{k-1} - \beta_k * \Delta_k \\ \hat{\theta}^k = \hat{\theta}^{k-1} - \dfrac{\alpha_k}{2\beta_k} * \mathcal{K}^k(\Delta_k) * (y_{2k} - y_{2k-1}) \end{cases} \tag{1.2}$$

$$\begin{cases} z^{2k} = \hat{\theta}^{k-1} + \beta_k * \Delta_k \\ z^{2k-1} = \hat{\theta}^{k-1} \\ \hat{\theta}^k = \theta^{k-1} - \dfrac{\alpha_k}{\beta_k} * \mathcal{K}^k(\Delta_k) * (y_{2k} - y_{2k-1}) \end{cases} \tag{1.3}$$

или даже с одним измерением:

$$\begin{cases} z^k = \hat{\theta}^{k-1} + \beta_k * \Delta_k \\ y_k = F(\omega^k, z^k) + \upsilon_k \\ \hat{\theta}^k = \hat{\theta}^{k-1} - \dfrac{\alpha_k}{\beta_k} * \mathcal{K}^k(\Delta_k) * y_k \end{cases} \tag{1.4}$$

где $\mathcal{K}(\cdot)$ – вектор-функции, удовлетворяющие вместе с функциями распределения пробного возмущения $P_k(\cdot)$ условиям:

$$\begin{cases} \int \mathcal{K}^k(s) P_k(ds) = 0 \\ \mathcal{K}^k(s) s^T P_k(ds) = 1 \\ \sup\limits_k \int \|\mathcal{K}^k(s)\|^2 P_k(ds) < \infty \end{cases}$$

Процедуры (1.2),(1.3), (1.4) называются ***рандомизированными алгоритмами стохастической аппроксимации***. Слово рандомизированный подчёркивает особую роль последовательности $\{\Delta_k\}$, элементы которой задают и новую точку измерения функции, и новую оценку точки минимума.

В англоязычной литературе для данных алгоритмов используется аббревиатура SPSA(Simultaneous Perturbation Stochastic Approximation).

Статья [24] приводит доказательство сходимости алгоритма при почти произвольных помехах в наблюдениях. В случае произвольных помех стоит скорее использовать (1.3), чем (1.2), что является следствием теоремы о сходимости из [24].

1.4.2 Сглаживаемый SPSA

В работе [5] была предложена новая версия SPSA, основанная на идее сглаживания градиента:

$$\begin{cases} \hat{\theta}^k = \hat{\theta}^{k-1} - \alpha_k * G_k \\ G_k = \rho_k G_{k-1} + (1-\rho_k)\hat{g}_k(\hat{\theta}^{k-1}) \\ G_0 = 0 \\ \rho_k \xrightarrow{k\to\infty} 0, \frac{\rho_k}{\alpha_k} = O(k^{-q}), q > 0 \end{cases} \tag{1.5}$$

где в качестве $\hat{g}_k(\hat{\theta}^{k-1})$ может использоваться одна из оценок градиента алгоритмов (1.2), (1.3), (1.4).

В этой же статье была доказана сходимость почти наверное последовательности оценок к оптимуму при тех же условиях, что и для несглаживаемой версии алгоритма. После рассмотрения двух численных примеров авторы делают вывод, что при тех же вычислительных затратах сглаживаемая версия в худшем случае работает не хуже, а в большинстве ситуаций лучше, чем обычная. Это можно объяснить тем, что сглаживание позволяет учитывать сразу несколько направлений поиска на первых итерациях.

1.4.3 SPSA второго порядка

Спалл [15] для ускорения сходимости алгоритма предложил рандомизированный алгоритм второго порядка(SPSA2), внешне похожий на известный метод Ньютона-Рафсона. SPSA2 требует всего 4 измерения функции за одну итерацию. Подход состоит из одновременного вычисления двух ре-

курсий:

$$\begin{cases} \hat{\theta}^k = \hat{\theta}^{k-1} - \alpha_k \bar{\bar{H}}_{k-1}^{-1} \hat{g}_k(\hat{\theta}^{k-1}), \bar{\bar{H}}_{k-1} = f_k(\bar{H}_k) \\ \bar{H}_{k-1} = \dfrac{k-1}{k} \bar{H}_{k-2} + \dfrac{1}{k} \hat{H}_{k-1} \end{cases} \tag{1.6}$$

где a_k – последовательность неотрицательных чисел, $\hat{g}_k(\hat{\theta}^{k-1})$ – оценка градиента в точке $\hat{\theta}^{k-1}$, $f_k : \mathbb{R}^{r \times r} \to \mathbb{R}^{r \times r}$ – преобразование для обработки ситуаций при отсутствии положительной определённости $\bar{H}_k$, $\hat{H}_{k-1}$ – оценка гессиана.

В статье проводится сравнение скорости сходимости рандомизированных алгоритмов первого(SPSA1) и второго порядков. В случае отсутствия шума в наблюдениях SPSA2 показывает гораздо более высокую скорость сходимости, чем SPSA1, при одинаковом количестве измерений. В ситуации с зашумлёнными данными SPSA2 превосходит SPSA1, но разница меньше, чем в первом случае.

Глава 2

Нейронные сети

2.1 Базовые понятия

Нейронной сетью называется вычислительная система, состоящая из простых, сильно взаимосвязанных элементов, накапливающих знания с помощью реакции на полученные данные и позволяющих создателю использовать эти знания.

Простейшим вычислительным элементом нейронной сети является **нейрон**(Рис. 2.1). Любой нейрон k описывается набором учитываемых сигналов x_j, каждому из которых поставлено в соответствие некоторое число w_{kj}, называемое **весом связи**, и функцией активации $\varphi(\cdot)$, применяемой к взвешенной линейной комбинации входных сигналов $\sum_{j=0}^{m} w_{kj}x_j$.

На данный момент разработано много различных архитектур нейронных сетей. Большинство из них предполагают, что нейроны объединяются в слои. Одной из популярных архитектур является **полносвязная многослойная сеть прямого распространения** (Рис. 2.2). Сигнал в такой сети распространяется от входного слоя к выходному, каждый нейрон следующего слоя имеет связи со всеми нейронами предыдущего, при этом связи между нейронами одного слоя отсутствуют.

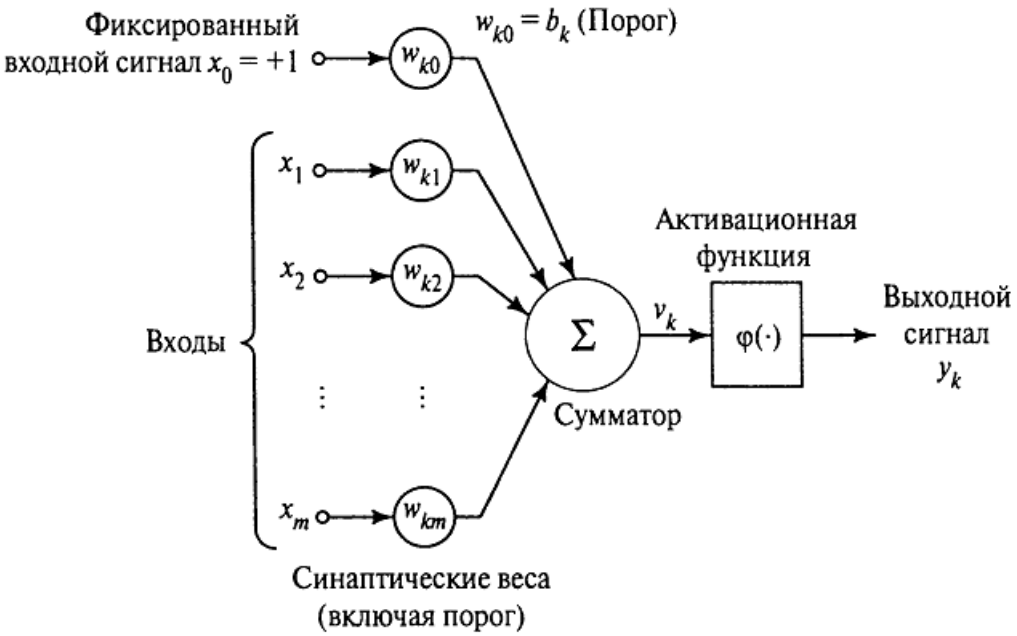

Рис. 2.1: модель нейрона
Источник: [26] с.45

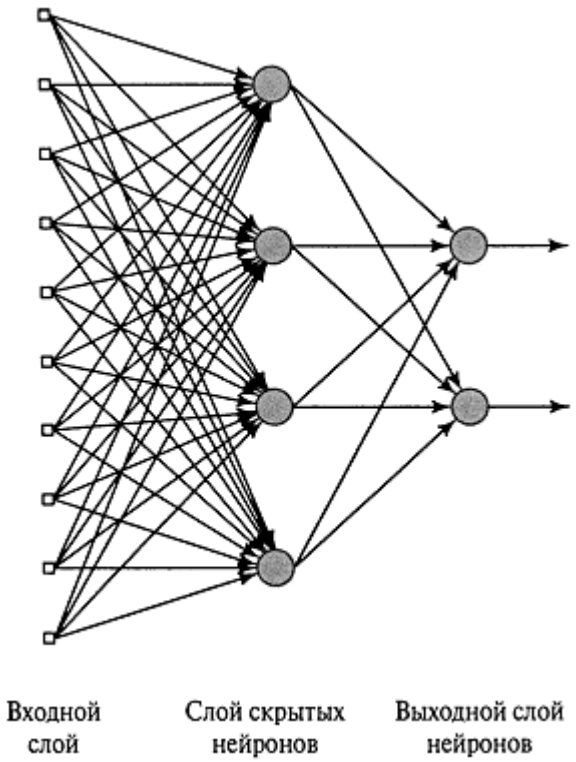

Рис. 2.2: полносвязная многослойная сеть прямого распространения
Источник: [26] с.57

2.2 Аппроксимация

Предполагается, что между входными данными и выходными существует некоторая функциональная зависимость

$$y = f(x)$$

за x обозначен вход, за y выход. Нейронной сети представляется набор примеров $\{(x_k, y_k)\}_{k=1}^{N}$, называемый **обучающим множеством**, на основе которого с помощью набора весов W строится отображение $\hat{f}(x)$. Для полученной функции должно иметь место:

$$\|\hat{f}(x_k) - f(x_k)\| \leq \varepsilon_1 \quad \forall k \tag{2.1}$$

для наперёд заданного ε_1. Главной задачей исследователей является осуществление условия:

$$\|\hat{f}(x) - f(x)\| \leq \varepsilon_2 \quad \forall x \in X \tag{2.2}$$

При решении данной задачи могут возникнуть следующие нежелательные ситуации:

1. недообучение, т.е. нарушение (2.1). Это соотношение будет верным только в том случае, когда правильно выбраны архитектура сети, вид функций активации $\varphi(\cdot)$ и имеется достаточное количество связей w_{kj}.

2. переобучение, т.е. выполнение (2.1) при нереализованном (2.2). Проблема возникает при слишком малом ε_1 или в случае недостаточной представительности обучающего множества $\{(x_k, y_k)\}_{k=1}^{N}$.

Целесообразность применения данного аппарата показывают теоретические результаты известной работы [22]. В этой статье было установлено, что любую непрерывную функцию можно представить с помощью нейронной сети с одной нелинейной функцией активации $\varphi(\cdot)$ сколь угодно точно. При этом [22] не содержит каких-либо теоретических результатов о требуемых количестве и размерах слоёв. В приложениях чаще всего используют 2 скрытых слоя.

2.3 Задачи управления

Работа [10] поспособствовала созданию теории управления, которая при постановке задач предполагает использование некоторой известной модели системы. Параметры модели подбираются на основе композиции априорной и апостериорной информации. С помощью нейронных сетей можно производить идентификацию систем, далее пользоваться уже известным аппаратом теории управления.

При помощи нейронных сетей возможно управлять разного рода системами и без использовании теории управления. Выделяют два подхода к обучению нейронных сетей в задачах управления: непрямой и прямой. Их отличие заключается в способе обновления свободных параметров сети.

2.3.1 Непрямое обучение

Непрямой подход предполагает построение отображения: данные о прошлом опыте и среде $\xrightarrow{W_1}$ управление u_k $\xrightarrow{W_2}$ реакция системы y_{k+1}. Изменяя веса W_2 на основе пар (u_k, y_{k+1}) строится зависимость между этими величинами, то есть аппроксимируется модель данной системы. После этого параметры W_2 фиксируются, далее происходит модификация W_1, используя W_2 для оценки частных производных.

Считая далее параметры W_2 неизменными, приходится сталкиваться с ошибкой, связанной с неизбежными немоделируемыми аспектами системы. Таким образом, управление, построенное на основе фиксированной модели, не может быть по-настоящему оптимальным.

2.3.2 Прямой подход

При таком обучении получается некоторая функция $\hat{f}$: данные о прошлом опыте и среде $\xrightarrow{W}$ управление u_k. Изменение весовых коэффициентов W напрямую зависит от выхода объекта управления y_k, а не через найденную приближенную модель системы, что и отличает данный подход от непрямого.

Постановка задачи

Пусть динамика системы описывается соотношением:

$$x_{k+1} = \phi_k(x_k, x_{k-1}, ..., x_{k-s}, u_k, w_k), s \geq 0 \tag{2.3}$$

где $\phi_k(\cdot)$ - неизвестная, в общем случае нелинейная функция, u_k - управление в момент k+1, $\{w_k\}$ - последовательность взаимно-независимых помех.

Нейронная сеть, описываемая вектором параметров θ_k, используется для аппроксимации управления $u_k = u_k(\theta_k; x_k, x_{k-1}, ..., x_{k-s}; t_{k+1})$, t_{k+1} - желаемый выход объекта управления в момент k+1. Необходимо найти такой вектор весов θ_k, чтобы значение следующего функционала было минимальным:

$$L_k(\theta_k) = E[(x_{k+1} - t_{k+1})^T A_k (x_{k+1} - t_{k+1}) + u_k^T B_k u_k] \tag{2.4}$$

где A_k, B_k - матрицы, задающие относительные веса отклонений от цели и затрат на управление. В данной работе рассмотрен случай, когда A_k - единичная, B_k - нулевая матрицы. Необходимо отметить, что функционал (2.4) описывает ошибку, полученную на одном шаге, а не на бесконечном интервале.

Часто в (2.3) удобно принимать $s = 0$ - случай Маркова.

Пошаговое описание алгоритма

1. На вход нейронной сети подаются текущее состояние системы x_k и следующее желаемое состояние t_{k+1}

2. Используя веса $\hat{\theta}_k$ нейронной сети, строится управление:
   $u_k = u_k(\hat{\theta}_k, x_k^+, t_{k+1})$

3. u_k подаётся на вход объекту управления, который переводит систему в x_{k+1} согласно (2.3)

4. Вычисляется мера эффективности: $\hat{L}_k = (x_{k+1} - t_{k+1})^T (x_{k+1} - t_{k+1})$

5. К весам добавляется пробное возмущение: $\hat{\theta}_k^+ = \hat{\theta}_k + \beta_k * \Delta_k$;

$\Delta_k = \begin{pmatrix} \Delta_{k1} \\ ... \\ \Delta_{kr} \end{pmatrix}$, где $\Delta_{ij} = \pm 1 \ \forall$ i,j с вероятностью 0.5 каждого значения

6. Аналогично пункту 2: $u_k^+ = u_k(\hat{\theta}_k^+, x_{k+1}^+, t_{k+1})$
7. Аналогично пункту 3 получено наблюдение x_{k+1}^+
8. Вычисляется мера эффективности: $\hat{L}_k^+ = (x_{k+1}^+ - t_{k+1})^T (x_{k+1}^+ - t_{k+1})$
9. Строится новая оценка $\hat{\theta}_{k+1} = \hat{\theta}_k - \frac{\alpha_k}{\beta_k}(\hat{L}_k^+ - \hat{L}_k)\Delta_k$

   $\alpha_k = \frac{\alpha}{(k+1+\xi)^\nu}$, $\beta_k = \frac{\beta}{(k+1)^\gamma}$
10. шаги 1-9 составляют одну итерацию алгоритма. Критерием остановки может служить количество итераций(или измерений), величина нормы разности оценок весов сети, величина разности мер эффективности.

Рассмотрение двух примеров

В первую модель управление входит мультипликативно, а шум - аддитивно.

$$x_{k+1} = (1 - (0.3 - 0.8e^{-x_k^2})x_k)u_k + w_k, w_k \sim N(0, \sigma^2) \tag{2.5}$$

где $x_k, u_k \in R^1$. Целевая последовательность $\{t_k\}$ является периодической: $t_k = t_{k+100n}$, первые 100 элементов задаются так:

$$\begin{cases} \sin(\frac{\pi k}{50}), & k \leq 50 \\ 1, & 51 \leq k \leq 75 \\ -1, & 76 \leq k \leq 100 \end{cases}$$

Во вторую модель управление входит аддитивно, шум мультипликативно.

$$x_{k+1} = \begin{pmatrix} -0.5 & 0.3 \\ 0 & 1.1 \end{pmatrix} x_k + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} u_k + \begin{pmatrix} \|x_k\| \\ 0 \end{pmatrix} w_k, \tag{2.6}$$

где $x_k, u_k \in R^2; t_k = (0,0) \ \forall k$.

Предлагается использовать полносвязную сеть прямого распространения с двумя скрытыми слоями. В качестве нелинейной функции активации используется гиперболический тангенс, в выходном слое применяется обычная линейная функция. Количество нейронов: 2-20-10-1 для (2.5), 4-20-10-2 для (2.6).

Запустим алгоритм для первой модели с параметрами: ($x_0 = 10$, $\alpha = 0.006$, $\xi = 10$, $\nu = 0.602$, $\beta = 0.25$, $\gamma = 0.101$, количество итераций $= 300$)

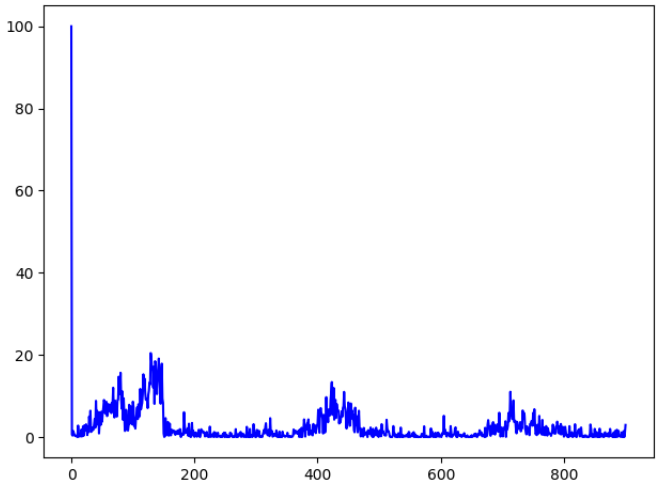

Рис. 2.3: Результат для первой модели

Запустим алгоритм для второй модели с параметрами: ($x_0 = [5, 5]$, $\alpha = 0.05$, $\xi = 0$, $\nu = 0.602$, $\beta = 0.25$, $\gamma = 0.101$, количество итераций $= 500$)

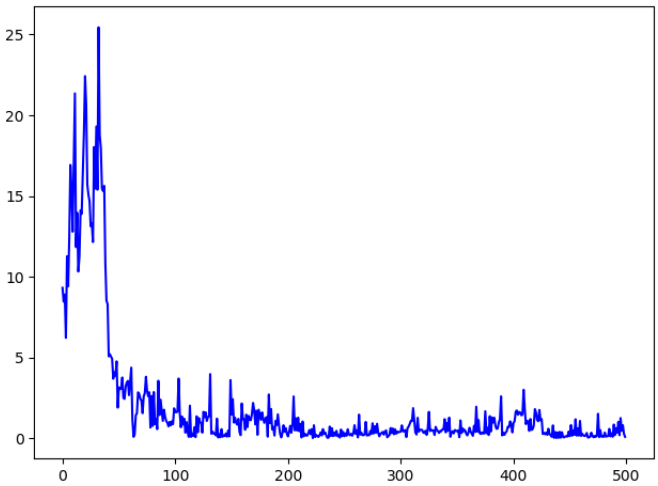

Рис. 2.4: Результат для второй модели

Глава 3

Оптимизация работы светофоров

3.1 Структура транспортной сети

Рассматривается система с 9 пересечениями. На каждом пересечении стоит светофор, который предоставляет проезд одному из двух несовместных направлений в течение первой фазы, другому на протяжении второй. Общее количество возможных пунктов отправления равно 16.

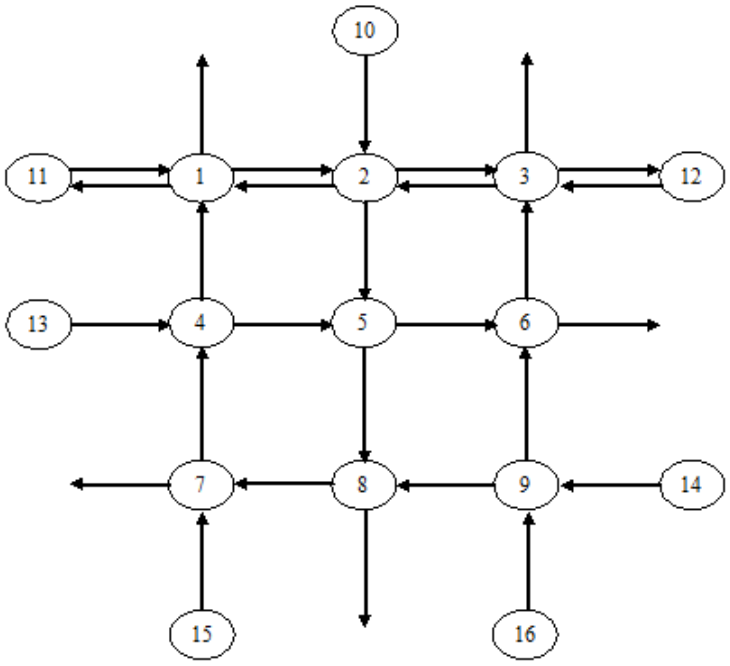

Рис. 3.1: Рассматриваемая сеть

Пунктом прибытия может быть любой, кроме $\{10, 13, 14, 15, 16\}$. Граф с Рис. 3.1 описывается при помощи матрицы смежности, задающей наличие рёбер и их направление.

$$E = \left(\begin{array}{cccccccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right) \mathbb{O}_{16\times 4}$$

Каждому ребру поставлены в соответствие его длина и ёмкость, задаваемые матрицами A_0 и C(см. приложения).

Итак, на выбор водителем маршрута влияют:

1. длины дуг A_0
2. пропускные способности дуг C
3. количество транспортных средств, предположительно использующих те же дуги
4. установки светофора L

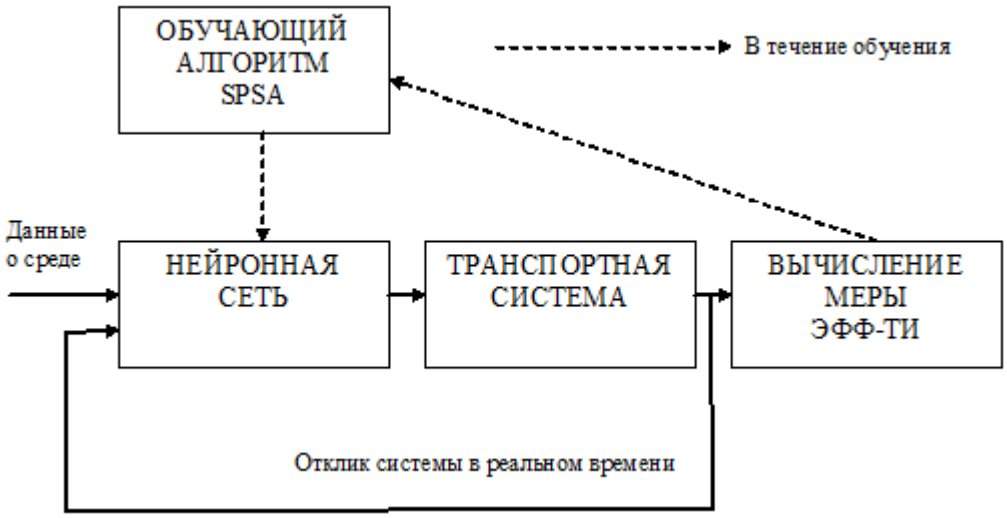

Рис. 3.2: Общая схема подхода

3.2 Метод решения

Метод заключается в построении функции управления $u(\cdot)$: данные о среде $\rightarrow$ оптимальные для сложившейся ситуации настройки сигналов светофоров. Данная функция реализуется с помощью нейронной сети, обучение которой происходит в рамках прямого подхода, описанного в подразделе 2.3.2.

Отличительной чертой метода является отсутствие модели движения транспорта, предполагается лишь само существование непрерывной зависимости между входными и выходными величинами. Это мотивируется тем, что даже если будет построена достаточно точная модель, она будет такой сложной, что её использование для быстрого поиска оптимальных настроек светофора будет невозможно. Так как не предполагается никакой модели системы, классический алгоритм обратного распространения ошибки неприменим в данной ситуации. Для прямого обучения используется алгоритм SPSA, вид которого подробно описан в главе 1, а его применение в задаче управления с помощью нейронной сети рассматривается в главе 2. Веса полностью определяют вид $u(\cdot)$, накапливая информацию о транспортном движении и устанавливая скрытые взаимосвязи между входными показателями и откликом системы. Следовательно, правильный подбор свободных параметров нейронной сети является главной целью данного подхода.

Предполагается, что разумным объёмом данных для обучения будет 90-дневный период, далее веса фиксируются, однако иногда можно “включать“ обучающий алгоритм для учёта сезонных изменений, делая 1-2 итерации. Нейронная сеть строится для определённого временного промежутка, например 17:00 – 19:00. Его длина должна быть такой, что распределение для каждой пары отправление-прибытие было примерно одинаковым в любой момент внутри этого промежутка. Если требуется иметь управление для более длительного окна, то следует поделить это окно на промежутки, отвечающие указанному свойству, далее для каждого из них построить своё управление. В ночное время 23:00 – 5:00 применение реагирующего управления не имеет смысла, поэтому для него можно использовать простую фиксированную стратегию.

Для того, чтобы проверить работоспособность алгоритма, обучение осуществляется на симулированных данных.

3.2.1 Моделирование

Объёмы движения

$$D = \begin{pmatrix} 0 & 17 & 250 & 10 & 378 & 265 & 222 & 68 & 0 & 0 & 94 & 399 \\ 340 & 0 & 481 & 408 & 190 & 83 & 53 & 436 & 0 & 0 & 344 & 323 \\ 380 & 191 & 0 & 122 & 285 & 302 & 482 & 291 & 0 & 0 & 92 & 190 \\ 373 & 384 & 293 & 0 & 38 & 132 & 2 & 276 & 0 & 0 & 184 & 407 \\ 196 & 399 & 112 & 175 & 0 & 328 & 389 & 72 & 0 & 0 & 314 & 267 \\ 329 & 93 & 377 & 98 & 266 & 0 & 410 & 428 & 0 & 0 & 391 & 176 \\ 85 & 245 & 128 & 126 & 391 & 375 & 0 & 312 & 0 & 0 & 40 & 471 \\ 354 & 223 & 253 & 309 & 468 & 226 & 42 & 0 & 0 & 0 & 466 & 439 \\ 15 & 324 & 350 & 237 & 65 & 42 & 200 & 257 & 0 & 0 & 389 & 276 \\ 139 & 356 & 447 & 176 & 285 & 114 & 130 & 201 & 0 & 0 & 244 & 312 \\ 23 & 378 & 481 & 417 & 235 & 458 & 401 & 38 & 0 & 0 & 0 & 294 \\ 48 & 138 & 274 & 293 & 5 & 76 & 216 & 120 & 0 & 0 & 224 & 0 \\ 413 & 341 & 69 & 275 & 169 & 414 & 457 & 61 & 0 & 0 & 153 & 151 \\ 348 & 328 & 74 & 460 & 81 & 270 & 91 & 92 & 48 & 0 & 255 & 236 \\ 159 & 81 & 129 & 143 & 398 & 500 & 132 & 120 & 0 & 0 & 256 & 115 \\ 477 & 59 & 422 & 380 & 156 & 39 & 73 & 209 & 472 & 0 & 410 & 423 \end{pmatrix} \mathbb{O}_{16\times 4}$$

Элемент $D(i,j)$ равен среднему ожидаемому значению объёма перемещений в паре (i,j) за один промежуток времени. Для каждого момента времени (p,t), где p – номер периода(дня), t – номер цикла внутри периода, задаётся матрица корреспонденций $D_{p,t}$. $\{D_{p,t}(i,j)\}$ - последовательность реализаций случайной величины, имеющей закон распределения Пуассона с параметром $\lambda = D(i,j)$.

Реакция транспортной системы

Для вычисления средних длин очередей используется программа [21], реализованная на языке Матлаб в виде функции с названием TAP. Теоретические основы для реализации [21] содержит работа [1]. Программа [21] содержит в себе заданные матрицы E, A_0, C, что позволяет моделировать движение в рассматриваемой транспортной сети.

Предполагается, что времени между моментами t и $t+1$ достаточно для того, чтобы все транспортные средства гарантированно успевают достигнуть заданных точек назначения.

На вход [21] получает матрицу корреспонденций $D_{p,t}$ и матрицу настроек сигналов $L_{p,t}$, полученной преобразованием из выхода нейронной сети $l_{p,t}$. Выходом программы является вектор средних длин очередей на каждой из дуг x за один цикл. Обобщая сказанное: программа используется для получения отображения $x = TAP(D_{p,t}, L_{p,t})$.

3.2.2 Архитектура нейроконтроллера

Предлагается использовать полносвязную нейронную сеть прямого распространения с двумя скрытыми слоями, так как такого количества слоёв обычно хватает для построения хорошего приближения. В качестве функции активации выступает гиперболический тангенс $\tanh(\upsilon) = \dfrac{e^{2\upsilon}-1}{e^{2\upsilon}+1}$ для обеспечения нелинейности преобразования.

На вход подаётся вектор $(d_{p,t}, l_{p,t-1}, t)^T$ размерности 152+9+1=162. $d_{p,t}$ – преобразованная в столбец матрица $D_{p,t}$, $l_{p,t-1}$ – столбец разбиений циклов светофоров, t - номер цикла с начала периода. Первый скрытый слой состоит из 34 нейронов, а второй из 12 нейронов. Выходной слой имеет 9 нейронов, каждый из которых соответствует одному из 9 светофоров, то

есть выходной вектор представляет собой $l_{p,t}$. Таким образом, необходимо оценить 163*34 + 35*12 + 13*9 = 6079 свободных параметров.

3.2.3 Пошаговое описание алгоритма

Инициализация $\hat{\theta}_0$: веса смещения для вычислительных элементов выходного слоя установить равными разбиениям циклов для фиксированной стратегии, например 0.45, все остальные установить равными 0. Если имеется ранее построенное управление при помощи нейронной сети такой же архитектуры, то свободные параметры можно скопировать из неё.

Чтобы получить новую оценку весов $\hat{\theta}_{k+1}$, необходимо:

1. в течение рассматриваемого периода дня использовать управление $u(\hat{\theta}_k, \cdot)$, где $\cdot$ обозначает входные данные $(d_{p,t}, l_{p,t-1}, t)^T$ для номера цикла t с начала периода

2. после каждого цикла измерять уровень очередей x и обновлять loss: loss $+= x^T x$

3. после того, как прошёл весь период, запомнить loss как y_-

4. Задаётся новое $\Delta_k = \begin{pmatrix} \Delta_{k1} \\ ... \\ \Delta_{kr} \end{pmatrix}$, где $\Delta_{ij} = \pm 1 \ \forall$ i,j с вероятностью 0.5 каждого значения

5. вычисляется $\beta_k = \dfrac{\beta}{(k+1)^\gamma}$

6. перед новым днём поменять значения весов на: $\hat{\theta}_k + \beta_k \Delta_k$; обнулить loss

7. повторить шаги 1,2 с весами $\hat{\theta}_k + \beta_k \Delta_k$

8. запомнить loss как y_+, обнулить loss

9. вычисляется $\alpha_k = \dfrac{\alpha}{(k+1+\xi)^\nu}$

10. обновить веса: $\hat{\theta}_{k+1} = \hat{\theta}_k - \dfrac{\alpha_k}{\beta_k}(y_+ - y_-)\Delta_k$

3.3 Запуски программы

Программа реализована в виде файла “spsa_2.m“. Матрицы корреспонденций генерируются в файле “PoissonData.m“. Внутри файла содержатся функции “spsa_2“, “vect2L“. Вторая из них нужна для преобразования $l_{p,t} \to L_{p,t}$, первая непосредственно реализует описанный подход.

“spsa_2“ получает на вход:

1. α: определяет величину изменения весов по итогам одной итерации, а, следовательно, и скорость сходимости алгоритма
2. β: влияет на размер пробного возмущения
3. ρ: отвечает за сглаживание градиента
4. γ: определяет скорость убывания последовательности β_k
5. ξ: стабилизирует алгоритм на первых шагах, позволяя увеличить α
6. ν: обеспечивает уменьшение α_k
7. ζ: задаёт скорость убывания ρ_k
8. h1_sz: размер первого скрытого слоя
9. h2_sz: размер второго скрытого слоя
10. days: количество периодов управления(дней) для обучения
11. moments: количество циклов в одном периоде управления

По оси абсцисс идут номера дней,по оси ординат значения сумм loss в течение дня. Синим показаны результаты применения фиксированной стратегии, красным – реагирующего управления.

Рассматриваются два случая: 90-дневный период для обучения с 30 циклами в день и 60-дневный период с 45 циклами в день. Пусть y_{test} - вектор, элементами которого являются дневные суммы длин очередей при фиксированной стратегии, y_{my} - при использовании рассмотренного подхода. Эффективность оценивается по двум параметрам:

1. пусть k - последний день, когда к весам не добавлялось пробное возмущение, тогда: $eff_1 = (y_{test}[k] - y_{my}[k])/y_{test}[k]$

2. $eff_2 = \dfrac{\sum_{s=1}^{days} y_{test}[s] - \sum_{s=1}^{days} y_{my}[s]}{\sum_{s=1}^{days} y_{test}[s]}$

eff_1 показывает, какой разницы в уровне эффективности стоит ожидать после отключения обучения. eff_2 характеризует разницу за время самого обучения. Решения стоит сравнивать в первую очередь по eff_1, так как при длительном использовании нейроконтроллера траты, описываемые eff_2, будут несущественными.

Для (0.05, 0.1, 0.5, 0.101, 0, 0.602, 0.603, 62, 24, 90, 30) получены пары: (0.074; 0.02), (0.115; 0.033), (0.1146; 0.0345), (0.1146; 0.0355), (0.1146; 0.0125). Отличия результатов объясняются использованием в SPSA случайных возмущений, некоторые из них быстро приводят к улучшению результатов, другие же ухудшают eff_2, но могут быть полезны на более поздних итерациях за счёт использования идеи "сглаживания". Усреднённый за 5 запусков график выглядит следующим образом: Точки, в которых значе-

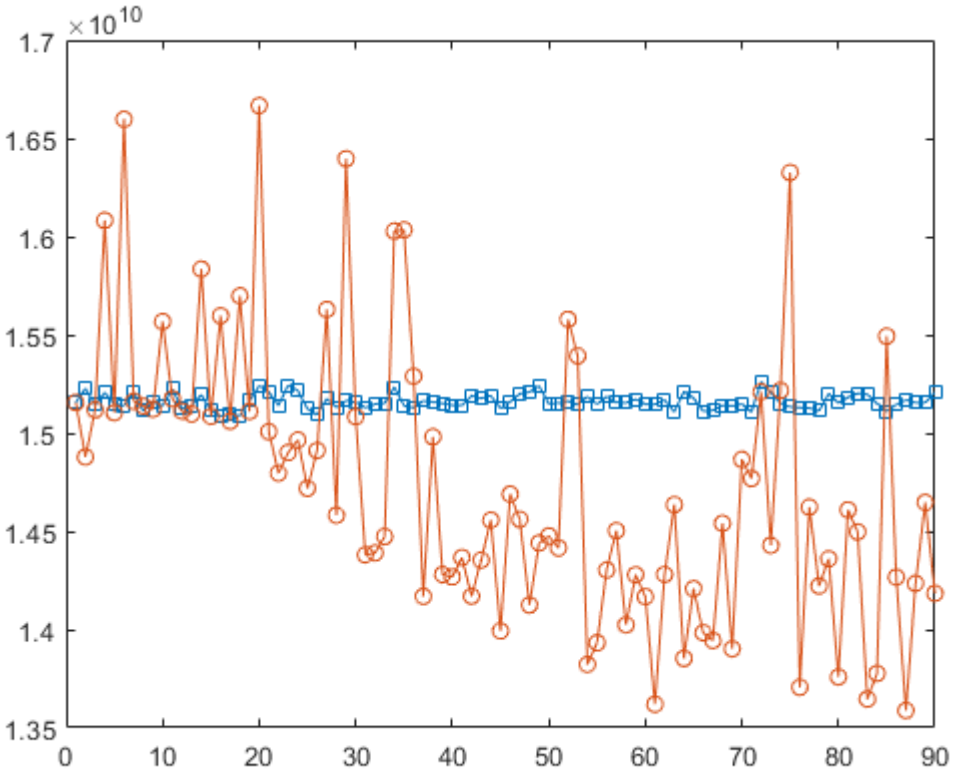

Рис. 3.3: Усреднённый график для 1-го набора

ния красного графика значительно выше соседних, обычно соответствуют

дням оценки после применения пробного возмущения. По графику отчётливо видно, что алгоритм пробует различные направления, чаще приводящих к негативным последствиям, однако данный опыт учитывается в новых оценках $\hat{\theta}_k$, которые соответствуют более успешным управлениям u_k. В среднем за 5 запусков алгоритм показал улучшение на 10,6 % .

Запуск программы для (0.04, 0.1, 0.5, 0.101, 0, 0.602, 0.603, 62, 24, 360, 30) показал, что 90 дней являются достаточным результатом для достижения оптимума, так как для 360-дневного промежутка был получен тот же минимум. Последовательности $\{\alpha_k\}$, $\{\beta_k\}$ со временем уменьшаются, что приводит к уменьшению колебаний графика в моменты возмущений и его стабилизации около значения $1.35 \cdot 10^{10}$.

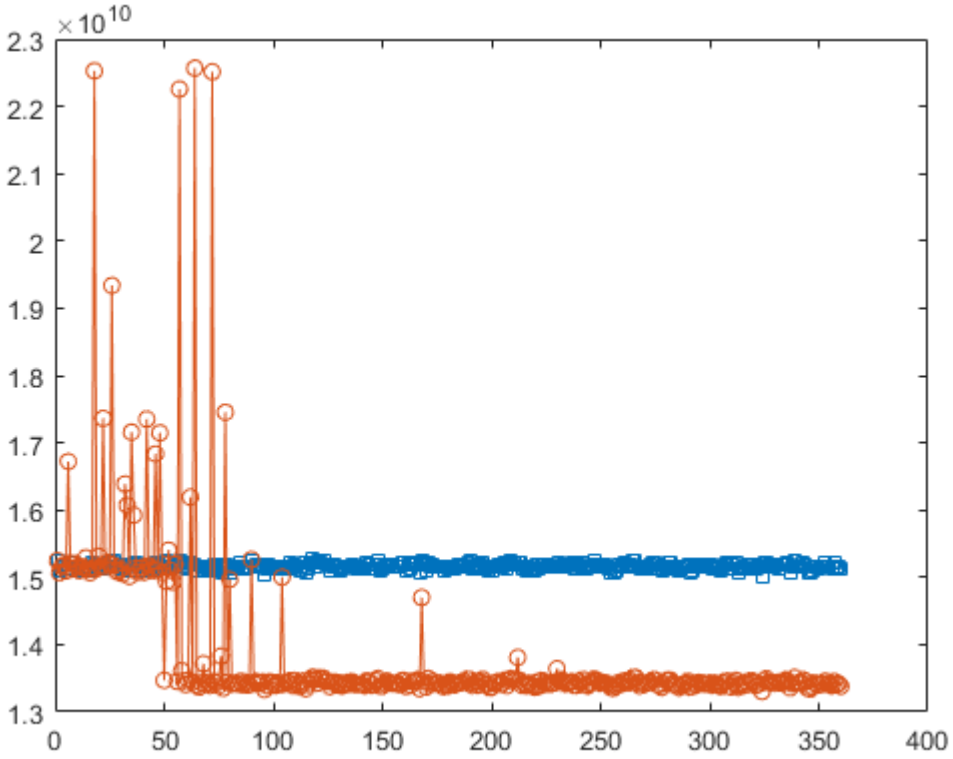

Рис. 3.4: 360 дней

Для случая 45 циклов в период амплитуда колебаний увеличивается, eff_2 для некоторых запусков может принимать отрицательные значения, в среднем для 10 запусков с (0.06, 0.11, 0.5, 0.101, 0, 0.602, 0.603, 62, 24, 60, 45) получается Рис.3.5. Установка $\beta = 0.11$ позволяет алгоритму активно исследовать пространство весов, однако такое большое изменение весов может приводить к резкому ухудшению состояния системы. При уменьшении

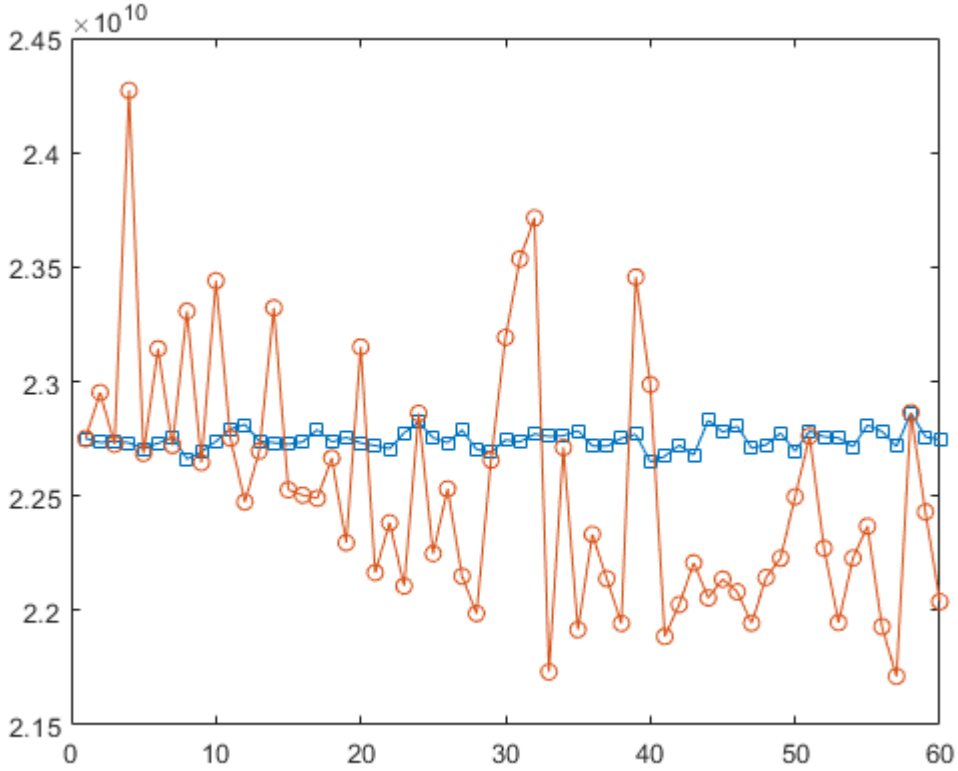

Рис. 3.5: 60 дней, 45 циклов

параметра β ниже 0.08 управление даёт низкую эффективность, примерно 0,2 – 0,5 %. Уменьшение в два-три раза количества нейронов в любом из скрытых слоёв также понижает эффективность до 0,3 – 0,8 %. Значительное уменьшение параметра α даже при сохранении β пределах 0.08 – 0.11 делает применение подхода нецелесообразным.

Выводы

В ходе запусков программы было установлено, что получаемая эффективность сильно зависит от значений α, β. Если эти параметры будут выбраны слишком большими, то веса сети значительно увеличатся, что приведёт к большим значениям сумм, для которых нейроны скрытых слоёв становятся равными ± 1. Если малыми, то построенный нейроконтроллер почти не будет отличаться от фиксированного управления. Классические рекомендации по подбору коэффициентов из статьи [13] не все оказались полезными для данной задачи.

Экспериментальным путём показано, что алгоритму не требуется большого количества итераций, чтобы дать достаточно весомое улучшение в 10%. Этот факт значительно повышает практическую ценность метода.

SPSA первого порядка не предполагает какого-либо масштабирования весовых коэффициентов, что может негативно сказываться при сильно отличающихся размерах слоёв. Возможным решением может стать использование улучшенного алгоритма второго порядка из [16].

Недостатком данной схемы является то, что алгоритм в качестве случайных направлений часто выбирает неудачные, однако их негативного влияния на следующие дни можно избежать при помощи блокирования таких итераций. Блокирование также позволяет увеличивать α, что ведёт к ускорению сходимости. Так как подход использует случайные возмущения, для одних и тех же входных данных получаются разные результаты, поэтому нельзя точно гарантировать нижнюю границу полезности данного способа.

Увеличение количества циклов в одном периоде приводит к увеличению чувствительности метода к входным параметрам и большей амплитуде сумм длин очередей.

Заключение

В ходе работы:

1. исследованы различные подходы к управлению сигналами светофоров, положение дел в данной области на текущий момент
2. изучены рандомизированные алгоритмы стохастической аппроксимации
3. проведено ознакомление с подходом управления, не требующего моделирования подконтрольной системы
4. рассмотрена схема применения SPSA для прямого обучения нейронной сети
5. разработана программная реализация алгоритма оптимизации режима работы светофоров для предложенной транспортной сети

Список литературы

[1] *A.Yu. Krylatov, A.P. Shirokolobova.* Projection approach versus gradient descent for network's flows assignment problem // Lecture Notes in Computer Science. 2017. 10556. 345–350.

[2] *Cheng X. J., Chang Q. S., Yang Z. X.* A traffic control system method based on Q-Learning // Syst. Eng. Theory Practice. 2006. 8. 136–140.

[3] *Chin D.C., Smith R.H.* A traffic simulation for mid-Manhattan with model–free adaptive signal control. 1994. 296—-301.

[4] *Chin D.C, Spall J.C., Smith R.H.* Evaluation of system-wide traffic signal control using stochastic optimization and neural network. San Diego, California, 1999. 2188—-2194.

[5] *Cristion J.A., Spall J.C.* Nonlinear adaptive control using neural networks: estimation with a smoothed form of simultaneous perturbation gradient // Statistica Sinica. 1994. 4. 1–27.

[6] *D.I. Robertson.* TRANSYT: A traffic network study tool // RRL Report. 1969.

[7] *Dell'Olmo P., Mirchandani P.* An approach for real-time coordination of traffic flows on networks // Transportation Research Board Annual Meeting. 1981. 950837.

[8] *Hockaday S. L., J. Martin P.* SCOOT—an update // ITE Journal. 1995. 65, 1. 44—-48.

[9] *Hunt P.B., Robertson D.I., Bretherton R.D., Winton R.I.* SCOOT-A traffic responsive method of coordinating signals // Transport and Road Research Laboratory. 1981.

[10] *Kalman R.E.* A new approach to linear filtering and prediction problems // Transactions of the ASME–Journal of Basic Engineering. 1960. 82. 35–45.

[11] *Kiefer J., Wolfowitz J.* Statistical estimation on the maximum of a regression function // Ann. Math. Statist. 1952. 23. 462–466.

[12] *Robbins H., Monro S.* A stochastic approximation method // Ann. Math. Statist. 1951. 22. 400–407.

[13] *Sadegh P., Spall J.C.* Optimal random perturbations for stochastic approximation using a simultaneous perturbation gradient approximation // IEEE Transactions on Automatic Control. 1998. 43, 10. 1480–1484.

[14] *Sims A. G.* The Sydney coordinated adaptive traffic system // Engineering Foundation Conference on Research Directions in Computer Control of Urban Traffic Systems, Pacific Grove, California, USA. 1979.

[15] *Spall J. C.* Accelerated second-order stochastic optimization using only function measurements // Proceedings of the 36th IEEE Conference on Decision and Control. 1997. 2. 1417–1424.

[16] *Spall J. C.* Adaptive stochastic approximation by the simultaneous perturbation method // IEEE transactions on automatic control. 2000. 45, 10. 1839–1853.

[17] *Spall J.C.* Multivariate stochastic approximation using a simultaneous perturbation gradient approximation // IEEE Transactions on Automatic Control. 1992. 37. 332–341.

[18] *Spall J.C., Chin D.C.* Traffic–responsive signal timing for system–wide traffic control // Transportation Research Part C. 1997. 5. 153–163.

[19] *Wiering M., Vreeken J., Veenen J., Koopman A.* A traffic control system method based on Q-Learning // IEEE Intell. Veh. Symp. 2004. 453–458.

[20] *Yang Y. P., Ou H. T.* Self-organized control of traffic signals based on reinforcement learning and genetic algorithm // Acta Autom. Sin. 2002. 28, 4. 564–568.

[21] *А.Ю. Крылатов.* Программа для равновесного распределения потоков по альтернативным маршрутам транспортной сети // Свидетельство о регистрации программы для ЭВМ RU 2018666089, 12.12.2018. Заявка № 2018663007 от 16.11.2018. ????

[22] *Горбань А.Н.* Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей // Сибирский журнал вычислительной математики. 1998. 1, 1. 11–24.

[23] *О.Н. Граничин.* Об одной стохастической рекуррентной процедуре при зависимых помехах в наблюдении, использующей на входе пробные возмущения // Вестник Ленинградского университета. Серия 1: Математика, механика, астрономия. 1989. 1. 19–21.

[24] *О.Н. Граничин.* Рандомизированные алгоритмы стохастической аппроксимации при почти произвольных помехах // Автоматика и телемеханика. 2002. 2. 44–55.

[25] *О.Н. Граничин, Б.Т. Поляк.* Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах. 2003.

[26] *С. Хайкин.* Нейронные сети: полный курс, 2-е издание. 2008.

Приложения

Программный код для моделей из главы 2

Приведена реализация подхода, описанного в главе 2, на языке программирования Python. Функция add_u_mult_w строит управление для второй модели, а mult_u_add_w – для первой модели.

Импортируются: библиотека numpy для использования матричных операций, реализации случайных величин, математических функций; фреймворк matplotlib.pyplot для построения графиков; из scipy.stats используется функция bernoulli для генерации шума в наблюдениях(для генерации с.в., распределённой по биномиальному закону, с возможностью задать p – вероятность успеха).

```
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import bernoulli


def my_norm(x):
    return np.sum(x**2) ** 0.5


def add_u_mult_w(x0, alpha, ksi, nju, beta, gamma, it):
    x = x0
    A = np.array([[-0.5, -0.3], [0, 1.1]])
    B = np.array([[0, 0], [0, 1]])
    C = np.array([[my_norm(x)], [0]])
    IH1_w =  np.random.sample((5, 20)) - 0.5
    H1H2_w = np.random.sample((21, 10)) - 0.5
    H2O_w =  np.random.sample((11, 2)) - 0.5
    in0 = np.array([x[0][0], x[1][0], 0, 0, 1], dtype=float)
    err = [0] * it
    n = 0
    for k in range(it):
        in1 = np.tanh(np.dot(in0, IH1_w))
```

```
        in2 = np.dot(np.hstack((in1, 1)), H1H2_w)
        in2 = np.tanh(in2)
        in3 = np.dot(np.hstack((in2, 1)), H2O_w)
        in3.shape = (2, 1)
        if k \% 2 == 0:
            w = bernoulli.rvs(0.5) - 0.5
            x = np.dot(A, x) + np.dot(B, in3) + C*w
            err[k] = my_norm(x)
            C[0][0] = err[k]
            in0[0], in0[1] = x[0][0], x[1][0]
            beta_n = beta/(n+1)**gamma
            delta1 = 2 * bernoulli.rvs(0.5, size=(5, 20)) - 1
            delta2 = 2 * bernoulli.rvs(0.5, size=(21, 10)) - 1
            delta3 = 2 * bernoulli.rvs(0.5, size=(11, 2)) - 1
            IH1_w += delta1 * beta_n
            H1H2_w += delta2 * beta_n
            H2O_w += delta3 * beta_n
        else:
            w = bernoulli.rvs(0.5) - 0.5
            x_plus = np.dot(A, x) + np.dot(B, in3) + C*w
            err[k] = my_norm(x_plus)
            alpha_n = alpha/(n+1+ksi)**nju
            mult = beta_n + alpha_n/beta_n * (err[k] - err[k-1])
            IH1_w -= delta1 * mult
            H1H2_w -= delta2 * mult
            H2O_w -= delta3 * mult
            n += 1
    arg = list(range(it))
    fig1, ax1 = plt.subplots()
    ax1.plot(arg, err, color="blue")
    plt.show()


def f(x, u):
    return (1 + (0.3 + 0.8*np.exp(-x**2))*x)*u + np.random.normal(0, 0.5)


def mult_u_add_w(x0, alpha, ksi, nju, beta, gamma, it):
    x = x0
    t = np.hstack((np.linspace(0, 2 * math.pi, 50), np.ones(25), -np.ones(25)))
    IH1_w = np.random.sample((3, 20)) - 0.5
    H1H2_w = np.random.sample((21, 10)) - 0.5
    H2O_w = np.random.sample((11, 1)) - 0.5
    in0 = np.array([x, t[0], 1], dtype=float)
    err = [0] * 3 * it
    n = 0
    for k in range(3 * it):
        if (k + 1) % 3 == 1:
            beta_n = beta / (n + 1) ** gamma
            delta1 = 2 * bernoulli.rvs(0.5, size=(3, 20)) - 1
```

```
            delta2 = 2 * bernoulli.rvs(0.5, size=(21, 10)) - 1
            delta3 = 2 * bernoulli.rvs(0.5, size=(11, 1)) - 1
            IH1_w -= delta1 * beta_n
            H1H2_w -= delta2 * beta_n
            H2O_w -= delta3 * beta_n
        elif (k + 1) % 3 == 2:
            IH1_w += 2 * delta1 * beta_n
            H1H2_w += 2 * delta2 * beta_n
            H2O_w += 2 * delta3 * beta_n
        else:
            alpha_n = alpha / (n + 1 + ksi) ** nju
            mult = beta_n + alpha_n * (err[k-1] - err[k-2]) / (2 * beta_n)
            IH1_w -= delta1 * mult
            H1H2_w -= delta2 * mult
            H2O_w -= delta3 * mult
            n += 1
            in0[1] = t[n % 100]
        in0[0] = x
        in1 = np.tanh(np.dot(in0, IH1_w))
        in2 = np.dot(np.hstack((in1, 1)), H1H2_w)
        in2 = np.tanh(in2)
        in3 = np.dot(np.hstack((in2, 1)), H2O_w)
        x = f(x, in3)
        err[k] = (x - in0[1]) ** 2
    err.insert(0, (x0-t[0]) ** 2)
    arg = list(range(3 * it + 1))
    fig1, ax1 = plt.subplots()
    ax1.plot(arg, err, color="blue")
    plt.show()


add_u_mult_w(np.array([[5], [5]]), 0.05, 0, 0.602, 0.25, 0.101, 500)
mult_u_add_w(10, 0.006, 10, 0.602, 0.04, 0.101, 300)
```

Матрицы длин и ёмкостей из главы 3

$$A_0 = \begin{pmatrix} 0 & 0.97 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.04 & 0 \\ 1.04 & 0 & 0.98 & 0 & 1.08 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.06 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.08 \\ 0.9 & 0 & 0 & 0 & 1.07 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.07 & 0 & 0.98 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.96 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.08 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.01 & 0 & 0.92 & 0 & 0 & 0 & 0 \\ 0 & 0.93 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.08 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.99 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.96 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.07 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.97 & 0 & 0 & 0 \end{pmatrix} \mathbb{O}_{16\times 4}$$

$$C = \begin{pmatrix} 0 & 63 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 329 & 0 \\ 325 & 0 & 386 & 0 & 414 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 458 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 276 \\ 438 & 0 & 0 & 0 & 132 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 158 & 0 & 62 & 0 & 0 & 0 & 0 \\ 0 & 0 & 310 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 450 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 125 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 271 & 0 & 492 & 0 & 0 & 0 & \\ 0 & 372 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 139 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 266 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 276 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 262 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 76 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 358 & 0 & 0 & 0 \end{pmatrix} \mathbb{O}_{16\times 4}$$

Программный код функций из главы 3

Используется язык программирования Matlab.

```
function [eff1, eff2] = spsa_2(alpha, beta, po, gamma, ksi, nju, zeta, h1_sz, h2_sz, days, mom
input_sz = 152 + 9 + 1;
out_sz = 9;
IH1_w = zeros(input_sz+1,h1_sz);
H1H2_w = zeros(h1_sz+1,h2_sz);
H2O_w = zeros(h2_sz+1,out_sz);
H2O_w(h2_sz+1,:) = 0.45;
G_IH1 = zeros(input_sz+1,h1_sz);
G_H1H2 = zeros(h1_sz+1,h2_sz);
G_H2O = zeros(h2_sz+1,out_sz);
load('poissonOD.mat')
D = zeros(16, 16);
D(:,:) = genD(:,:,1,1);
[coord1, coord2] = find(D);
d = zeros(1, 152);
in3 = 0.45 * ones(1, 9);
n = 0;
flag = 0;
y_my = zeros(1, days);
L_test = vect2L(in3);
y_test_p = zeros(1, days);
check = 1;
for day=1:days
    loss = 0;
    for hour=1:moments
        D(:,:) = genD(:,:,day,hour);
        for k=1:152
            d(k) = D(coord1(k), coord2(k));
        end
        in0 = [in3, d/1300, (hour-1)/(moments-1), 1];
        in1 = in0 * IH1_w;
        in1 = [tanh(in1), 1];
        in2 = in1 * H1H2_w;
        in2 = [tanh(in2), 1];
        in3 = in2 * H2O_w;
        in3(in3 < 0.05) = 0.05;
        in3(in3 > 0.95) = 0.95;
        TAP(D, L_test)
        load('raspr.mat')
        y_test_p(day) = y_test_p(day) + sum(x1.*x1);
        TAP(D, vect2L(in3))
        load('raspr.mat')
        x1 = x1([1,3:6,8:23]);
        loss = loss + sum(x1.*x1);
    end
    y_my(day) = loss;
```

```
        if flag == 0
            if (check == 0) && (y_my(day) > 1.01 * y_my(day-2))
                IH1_w = IH1_w + alpha_n * G_IH1;
                H1H2_w = H1H2_w + alpha_n * G_H1H2;
                H2O_w  = H2O_w + alpha_n * G_H2O;
                G_IH1  = G_IH1  - 0.9 * mult * delta1;
                G_H1H2 = G_H1H2 - 0.9 * mult * delta2;
                G_H2O  = G_H2O  - 0.9 * mult * delta3;
                check = 1;
            else
                flag = 1;
                y_minus = loss / 10^9;
                delta1 = 2*round(rand(input_sz+1,h1_sz)) - 1;
                delta2 = 2*round(rand(h1_sz+1, h2_sz)) - 1;
                delta3 = 2*round(rand(h2_sz+1, out_sz)) - 1;
                beta_n = beta/(n+1)^gamma;
                IH1_w  = IH1_w  + beta_n * delta1;
                H1H2_w = H1H2_w + beta_n * delta2;
                H2O_w  = H2O_w  + beta_n * delta3;
            end
        else
            flag = 0;
            check = 0;
            y_plus = loss / 10^9;
            alpha_n = alpha/(n+1+ksi)^nju;
            po_n = po/(n+1)^zeta;
            mult = (1-po_n) * (y_plus - y_minus)/beta_n;
            if abs(mult * alpha_n) > beta_n / 2
                mult = sign(mult) * beta_n / 2 / alpha_n;
            end
            G_IH1  = po_n * G_IH1  + mult * delta1;
            G_H1H2 = po_n * G_H1H2 + mult * delta2;
            G_H2O  = po_n * G_H2O  + mult * delta3;
            IH1_w  = IH1_w -  beta_n * delta1 - alpha_n * G_IH1;
            H1H2_w = H1H2_w - beta_n * delta2 - alpha_n * G_H1H2;
            H2O_w  = H2O_w -  beta_n * delta3 - alpha_n * G_H2O;
            n = n + 1;
        end
    end
    plot(1:days, y_test_p,'-s', 1:days, y_my, '-o')
    if flag == 1
        k = days;
    elseif check == 0
        k = days - 1;
    else
        k = days - 2;
    end
    eff1 = (y_test_p(k) - y_my(k)) / (y_test_p(k));
    eff2 = (sum(y_test_p) - sum(y_my)) / sum(y_test_p);
end
```

```
97
98  function matr = vect2L(L_v)
99      matr = zeros(16,16);
100     matr(11,1) = L_v(1);
101     matr(2,1) = matr(11,1);
102     matr(4,1) = 1 - matr(2,1);
103     matr(1,2) = L_v(2);
104     matr(3,2) = matr(1,2);
105     matr(10,2) = 1 - matr(3,2);
106     matr(2,3) = L_v(3);
107     matr(12,3) = matr(2,3);
108     matr(6,3) = 1 - matr(12,3);
109     matr(13,4) = L_v(4);
110     matr(7,4) = 1 - matr(13,4);
111     matr(4,5) = L_v(5);
112     matr(2,5) = 1 - matr(4,5);
113     matr(5,6) = L_v(6);
114     matr(9,6) = 1 - matr(5,6);
115     matr(8,7) = L_v(7);
116     matr(15,7) = 1 - matr(8,7);
117     matr(9,8) = L_v(8);
118     matr(5,8) = 1 - matr(9,8);
119     matr(14,9) = L_v(9);
120     matr(16,9) = 1 - matr(14,9);
121 end
122
123 function PoissonData(days_number, hours_number)
124     D = [
125                 0    17   250     10   378   265   222    68     0     0    94   399     0     0
126                 340     0   481   408   190    83    53   436     0     0   344   323     0     0
127                 380   191     0   122   285   302   482   291     0     0    92   190     0     0
128                 373   384   293     0    38   132     2   276     0     0   184   407     0     0
129                 196   399   112   175     0   328   389    72     0     0   314   267     0     0
130                 329    93   377    98   266     0   410   428     0     0   391   176     0     0
131                 85   245   128   126   391   375     0   312     0     0    40   471     0     0
132                 354   223   253   309   468   226    42     0     0     0   466   439     0     0
133                 15   324   350   237    65    42   200   257     0     0   389   276     0     0
134                 139   356   447   176   285   114   130   201     0     0   244   312     0     0
135                 23   378   481   417   235   458   401    38     0     0     0   294     0     0
136                 48   138   274   293     5    76   216   120     0     0   224     0     0     0
137                 413   341    69   275   169   414   457    61     0     0   153   151     0     0
138                 348   328    74   460    81   270    91    92    48     0   255   236     0     0
139                 159    81   129   143   398   500   132   120     0     0   256   115     0     0
140                 477    59   422   380   156    39    73   209   472     0   410   423     0     0
141             ];
142     genD = zeros(16, 16, days_number, hours_number);
143     for l=1:16
144         for c=1:16
145             if D(l,c) ~= 0
146                 genD(l, c, :, :) = poissrnd(D(l,c), days_number, hours_number);
```

```
147                 end
148             end
149         end
150 save poissonOD genD
```